IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	10/795,923	Confirmation No. 1818
Inventor	:	Kegel	
Filed	:	03/08/2004	
TC/A.U.	:	2143	
Examiner	:	Jude Jean-Gilles	
Docket No.	:	I004-P03074US	
Customer No.	:	33356	

APPEAL BRIEF

The following Appeal Brief is submitted pursuant to the Notice of Appeal filed June 12, 2008 for consideration by the Board of Appeals and Interferences. 37 C.F.R. § 41.37.

(i) REAL PARTY IN INTEREST

The real party in interest is Ixia.

(ii) RELATED APPEALS AND INTERFERENCES

There are no appeals, interferences or judicial proceedings which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(iii) STATUS OF CLAIMS

Claims 1-35 were pending and rejected in the Final Office Action dated March 12, 2008. Claims 1-35 are pending and are the subject of this appeal.

(iv) STATUS OF AMENDMENTS

No amendments to the claims were filed after the Final Office Action dated March 12, 2008.

(v) SUMMARY OF CLAIMED SUBJECT MATTER[1]

Claim 1

A method of creating network traffic (paras. 0023 and 0024) replicating activities of a large number of users (paras. 0027) comprising:

receiving a test script including a plurality of commands (FIG. 5, bock 510; paras. 0027, 0038, 0039 and 0041)

invoking a script interpreter (210, 310 and 410; FIG. 5, block 512; paras. 0028, 0038, 0039 and 0041)

launching an application thread (para. 0032) to execute the test script (220, 320, 420; FIG. 5, block 514; paras. 0034, 0038, 0039 and 0041)

invoking a protocol engine (230, 330, 430) for each of the commands in the test script such that each protocol engine has an associated command, (FIG. 5, block 516, paras. 0034, 0038, 0039 and 0041)

each protocol engine executing its associated command. (FIG. 5, block 530, para. 0044)

Claim 2

The method of claim 1 wherein the commands in the test script simulate actions taken by a network user. (paras. 0027 and 0029)

Claim 5

The method of claim 1 wherein the test script causes network traffic to be produced. (paras. 0026 and 0027)

Claim 11

A machine readable medium (para. 0019) having instructions (para. 0018) stored thereon which when executed cause a processor (para. 0014) to perform operations comprising:

[1] All references herein are to the application and drawings as originally filed, pursuant to 37 C.F.R. § 41.37(c)(1)(v).

receiving a test script including a plurality of commands (FIG. 5, bock 510; paras. 0027, 0038, 0039 and 0041)

invoking a script interpreter (210, 310 and 410; FIG. 5, block 512; paras. 0028, 0038, 0039 and 0041)

launching an application thread (para. 0032) to execute the test script(220, 320, 420; FIG. 5, block 514; paras. 0034, 0038, 0039 and 0041)

invoking a protocol engine (230, 330, 430) for each of the commands in the test script such that each protocol engine has an associated command, (FIG. 5, block 516, paras. 0034, 0038, 0039 and 0041)

each protocol engine executing its associated command. (FIG. 5, block 530, para. 0044)

Claim 12

The machine readable medium (para. 0019) of claim 11 wherein the commands in the test script simulate actions taken by a network user. (paras. 0027 and 0029)

Claim 15

The machine readable medium (para. 0019) of claim 11 wherein the test script causes network traffic to be produced. (paras. 0026 and 0027)

Claim 21

A system (110) to create network traffic (paras. 0023 and 0024) simulating activities of a large number of users (para. 0027), the system comprising:

a plurality of script interpreter units (210) in user space (paras. 0028, 0034, and 0037), each script interpreter unit to interpret a script including a plurality of commands, (paras. 0027 and 0028)

an application thread (220, para. 0032) in user space for each script interpreter unit (paras. 0034, 0037) a plurality of protocol engines (230) in user space for each application thread, (paras. 0034 and 0037) each protocol engine to execute a command included in one of the scripts (para. 0034)

an operating system (250) in operating system space (260). (paras. 0034 and 0037)

<u>Claim 26</u>

A system (110) to create network traffic (paras. 0023 and 0024) simulating activities of a large number of users (para. 0027), the system comprising:

a plurality of script interpreter units (310) in user space (304), each script interpreter unit to interpret a script including a plurality of commands, (paras. 0028, 0034, and 0038),

an application thread (320) in user space (304) for each script interpreter unit (para. 0038)

a plurality of protocol engines (330) in user space (304) for each application thread (320), each protocol engine (330) to execute a command included in one of the scripts (para. 0038)

an operating system (350) in operating system space (360). (para. 0038)

<u>Claim 31</u>

A system (110) to create network traffic (paras. 0023 and 0024) simulating activities of a large number of users (para. 0027), the system comprising:

a plurality of script interpreter units (410) in user space (404), each script interpreter unit to interpret a script including a plurality of commands, (paras. 0028, 0034 and 0039)

an application thread (420) in operating system space (460) for each script interpreter unit (410) (para. 0039) a plurality of protocol engines (430) in operating system space (460) for each application thread (420), each protocol engine (430) to execute a command included in one of the scripts (para. 0039)

an operating system (450) in operating system space (460). (para. 0039)

(vi) GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The Examiner rejected claims 1-2, 5-12, 15-23, 26-28 and 31-33 under 35 USC § 103 as obvious from Smith et al. (US Patent No. 6,091,802) in view of Averbuj (US Patent Publication No. 20050257109 A1).

The Examiner rejected claims 3-4, 13-14, 24-25, 29-30 and 34-35 under 35 USC § 103 as obvious from Smith in view of Averbuj and Jameson (US Patent Publication No. 20030107596 A1).

(vii) ARGUMENT

A. All Claims: The References May Not Properly Be Combined

Claims 1-2, 5-12, 15-23, 26-28 and 31-33 were rejected as obvious in view of Smith and Averbuj. Claims 3-4, 13-14, 24-25, 29-30 and 34-35 were rejected as obvious in view of Smith, Averbuj and Jameson. We assert that these rejections are not proper because the cited references may not properly be combined.

As set forth in their titles and abstracts, Smith is directed to a Telecommunication System Tester with Integrated Voice and Data that "includes a test computer for scheduling and controlling the execution of test scripts ... to simulate transactions that typically take place on the telecommunications system", and Averbuj is directed to a Built-In Self Test (BIST) Architecture that "stores a set of commands that generically define an algorithm for testing memory modules" in a single device. The references are so unrelated that it would not be logical to combine different aspects to result in the claimed subject matter.

Succinctly, Smith discloses a telecommunications testing system and Averbuj discloses a memory testing system. These publications disclose different techniques that solve problems that are wholly unrelated to one another. Stated another way, Smith teaches a computer that performs telecommunications testing over a telecommunications network. Averbuj teaches a controller that performs memory testing of a single electronic device. The low level, chip based testing techniques of Averbuj exist in a different realm and are not applicable to the higher level, inter-device and telecommunications network testing techniques taught in Smith. Simply, Smith and Averbuj are in different technology spaces. As such, their teachings may not be properly combined.

The Examiner states that Averbuj "teaches command protocols that allow powerful algorithms or test scripts running on distributed memory modules". (Final Office Action, p. 3, lines 16-17) But the teachings in Averbuj involve memory modules distributed within a single device. Averbuj teaches that the BIST controller distributes test algorithms to memory modules included in a single electronic device in which the memory and BIST controller reside. (Averbuj, para. 0008 et seq.) The improvement of Averbuj is a central BIST controller for all memory modules in a single electronic device. (Averbuj, para. 0015) Averbuj is limited to testing memory included in a single device. These teachings make it abundantly clear that Averbuj in no way involves the area of telecommunications among multiple devices of Smith, but is limited to the niche of memory testing on an electronic device.

Further, Smith and Averbuj are in wholly different patent classes, and, as such, may not be properly combined. Smith is in class 379/29 and various other subclasses in class 379. Class 379 is for "telephonic communications" while subclass 29.01 is more specifically directed to "terminal arrangement to enable remote testing (e.g., testing interface)". Differently, Averbuj was published in class 714/33 which is directed to "error detection/correction and fault detection/recovery" "derived from analysis (e.g., of a specification or by stimulation)". Now in prosecution, Averbuj has been placed in class 714/718. Class 714/718 is specifically directed to "memory testing". That the U.S.P.T.O.'s classification system shows that Smith and Averbuj are in wholly different patent classes supports the conclusion that the references may not be properly combined.

In addition, it is not apparent why a person involved with a telecommunications testing system of Smith would avail himself of the memory testing system of Averbuj. "[T]he Examiner has not provided a sufficient reason or explicit analysis of why the disclosures of the references should be combined." *Ex parte Erkey at al.*, Appeal 20071375 (BPAI May 11, 2007).

For example, in discussing claim 1, the Examiner asserts that the first three steps of the method are disclosed in Smith, while the last two steps are asserted to be disclosed in Averbuj. However, there is no teaching or suggestion in Averbuj of telecommunications, and there is no teaching or suggestion in Averbuj of the kind of network protocols as disclosed and claimed in Smith. ("The specified network protocol may typically be a loop-start protocol, a wink-start protocol, or a MFC-R2". Smith 5:54-56) As such, there is no tie between the memory testing system of Averbuj and the telecommunications testing system of Smith.

Last, Smith discloses that it uses test scripts that include commands involving telephony functions. (See Smith, 5:56-61) There is no problem in Smith's use of test scripts; the scripts achieve their intended purpose. In the *KSR* case, the Supreme Court qualified the issue of hindsight by stating that "[r]igid preventative rules that deny factfinders recourse to common sense, however, are neither necessary under our case law nor consistent with it." *KSR Int'l Co. v. Teleflex, Inc.*, 127 S.Ct. 1727 (2007). In the instant matter, a person of ordinary skill in the art having common sense at the time of the invention would not have reasonably looked to Averbuj to solve a problem with scripts or test algorithms, because Smith did not have any such problems to be solved. See *Ex parte Rinkevich et al.*, Appeal 20061317 (BPAI May 29, 2007). It can only be from improper hindsight that the Examiner is attempting to combine Smith and Averbuj. Since Smith already teaches a solution, it is clear that the Examiner has used the claims as a guide or roadmap in formulating the rejection. As such, the references may not be properly combined to make an obviousness rejection.

As set forth above, the rejection of all claims based on the cited references is not well founded and should be withdrawn.

B. All Claims: The References Do Not Disclose The Claimed "Protocol Engine"

The independent claims, claims 1, 11, 21, 26 and 31, recite a "protocol engine". The Examiner directs us to Averbuj for a teaching of protocol engines. The Examiner asserts that the sequencers and test algorithms of Averbuj teach the claimed protocol engine. However, the sequencers and test algorithms of Averbuj fail to teach or suggest the protocol engine claimed.

The term "protocol engine" must be interpreted in view of the entirety of the claim as well as the specification. Claim 1 recites a "method of creating network traffic", while claims 21, 26 and 31 each recite a "system to create network traffic". Further, claims 1 and 11 recite "invoking a protocol engine for each of the commands in the test script", and claims 21, 26 and 31 recite "each protocol engine to execute a command included in one of the scripts". As such, the "protocol engine" necessarily involves "network traffic" and "test scripts". Moreover, the entirety of the specification involves network traffic and test scripts that are used to create network traffic. This cannot be ignored in interpreting the claimed "protocol engine".

As to "test scripts", the specification states the following regarding test scripts.

> A **test script** may include a sequence of commands or instructions that cause **network traffic** to be created and transmitted by the network testing system. Each **test script** may replicate the **network traffic** generated by a single network user or a group of network users. Each test script may represent or be a simulated virtual user or a group of simulated virtual users. Each test script may be used to stress test a device under test and/or a portion of a network and/or a network. Such a portion of a network or network may include one or more devices under test.
>
> Specification, para. 0027 (**emphasis** added)

As to "network traffic", the specification defines "network traffic" as "data units communicated over a network" which supports communications according to one or more higher level or lower level communications protocols such as UDP, TCP, FTP, ISDN, PPP, FDDI and others. (Specification, paras. 0013, 0023 and 0024)

The Examiner asserts that the sequencers (element 8) of Averbuj teach claimed "protocol engine". This cannot be so. Averbuj states that

> Sequencers 8 interpret and execute test algorithms provided by BIST controller 4. In particular, sequencers 8 receive high-level commands from BIST controller 4 that define a complete BIST algorithm.
>
> Averbuj, para. 0030, first 4 lines.

All this portion of Averbuj teaches is that a sequence of commands may be included in a test algorithm for testing memory in a device. There is no teaching or suggestion in Averbuj of "network traffic". It follows that there is no teaching in Averbuj of test scripts and scripts that include commands involving network communications. Thus, Averbuj fails to teach a "protocol engine" as claimed.

Because this limitation is not found in the cited reference, the independent claims are patentable over the cited references. The independent claims are patentable over the combination of cited references because the combination of cited references do not teach or suggest all of the limitations recited.

C. Claims 2 and 12: The Combination of References Can Not Be Cited For Teaching Simulating Actions Taken by a Network User

Claims 2 and 12 recite that "the commands in the test script simulate actions taken by a network user". We assert that Smith cannot be combined with Averbuj to teach "the commands in the test script simulate actions taken by a network user".

The Examiner directs us to Smith for the teaching of simulating user actions with a telecommunications system. (Smith, col. 2, lines 19-32; col. 3, lines 30-38; Final Office Action, p. 8, lines 3-4) We agree that Smith teaches simulating the actions of a user. However, the use of this teaching in the system constructed by the Examiner is impossible for the reasons set forth in the following paragraphs.

Claims 2 and 12 depend on claims 1 and 11. Claims 1 and 11 recite that the claimed protocol engine is invoked for each of the commands in the test script. As such, in view of the totality of claims 2 and 12, the protocol engine invokes commands from the test script to "simulate actions taken by a network user". However, when addressing claim 1, the Examiner states that "Smith does not specifically disclose 'invoking a protocol engine for each of the commands in the test script such that each protocol engine has an associated command', and 'each protocol engine executing its associated command'". (Final Office Action, p. 6, bottom 3 lines)

So what we have here is that on one page of the Final Office Action the Examiner states that Smith does not disclose the protocol engine, but then on another page the Examiner asserts that simulating users which is achieved by the protocol engines is taught by Smith. This does not make logical sense.

We assert that Smith cannot be cited for disclosing that the script invoked by the protocol engine simulates the actions taken by a network user, because, as stated by the Examiner, there is no protocol engine disclosed in Smith.

In addition, we assert that the chip testing system of Averbuj cannot teach this limitation because Averbuj does not in any way involve a network or actions taken by a network user. That is, the sequencers of the memory chip testing system of Averbuj (which the Examiner asserts teach the protocol engines) are not capable of performing actions that "simulate actions taken by a network user".

Because the components cited by the Examiner in the cited references are incapable of performing actions that "simulate actions taken by a network user", the references do not teach the limitations for which they are cited. As such, claims 2 and 12 are patentable over the cited references.

D. Claims 5 and 15: The Combination of References Can Not Be Cited For Teaching Test Scripts That Cause Network Traffic To Be Produced

Claims 5 and 15 recite that "the test script causes network traffic to be produced". We assert that Smith cannot be combined with Averbuj to teach this limitation.

Claims 5 and 15 depend on claims 1 and 11. Claims 1 and 11 recite that the claimed protocol engine is invoked for each of the commands in the test script. As such, in view of the totality of claims 5 and 15, the protocol engine invokes commands from the test script to cause network traffic to be produced. However, when addressing claim 1, the Examiner states that "Smith does not specifically disclose 'invoking a protocol engine for each of the commands in the test script such that each protocol engine has an associated command', and 'each protocol engine executing its associated command'". (Final Office Action, p. 6, bottom 3 lines) The Examiner then asserts that Smith teaches that the test script causes network traffic to be produce. (Final Office Action, p. 8, lines 5-8) That is, on one page of the Final Office Action the Examiner states that Smith does not disclose the protocol engine, but then on another page the Examiner asserts that network traffic produced by the protocol engine is taught by Smith. This does not make logical sense.

We assert that Smith cannot be cited for disclosing that the script invoked by the protocol engine produces network traffic, because, as stated by the Examiner, there is no protocol engine disclosed in Smith.

The Examiner asserts that Smith teaches the claimed network traffic that is produced by the protocol sequencers of Averbuj. But this cannot be, as the protocol sequencers that perform the chip testing teachings of Averbuj have no network communications capabilities whatsoever.

Because the components cited by the Examiner in the cited references are incapable of teaching a test script that causes network traffic to be produced, the references do not teach the limitations for which they are cited. As such, claims 5 and 15 are patentable over the cited references.

Further, we assert that the chip testing system of Averbuj cannot teach this limitation because Averbuj does not disclose network communications and network traffic. The specification defines "network traffic" as "data units communicated over a network" which supports communications according to one or more higher level or lower level communications protocols such as UDP, TCP, FTP, ISDN, PPP, FDDI and others. (Specification, paras. 0013, 0023 and 0024) Because "network traffic" as defined in the specification and used in the claims is not found in the cited references, claims 5 and 15 are patentable over the cited references.

E. All Claims: The References Do Not Disclose The Claimed "Application Thread"

The independent claims recite an "application thread". "Importantly, the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification." *Philips v AWH* 415 F.3d 1314, (Fed. Cir. 2005) (*en banc*) As set forth in the specification, an application thread is not a traditional operating system thread as it is lighter weight such that it requires "a smaller amount of network testing system resources to execute". (Specification, para. 0033)

The Examiner directs us to col. 4, lines 61-65 of Smith for the teaching of an application thread. However, there is nothing in Smith that discloses application threads. Smith discusses "threads of control" that may not exceed the number of available processors. (Smith, 4:61-65) Smith states that the threads may be those created under Microsoft Windows NT. (Smith, 4:46-50) As such, the threads taught in Smith are traditional operating system threads. In this way, the "threads of control" of Smith do not teach or suggest the claimed application threads. Averbuj fails to disclose an application thread.

The Examiner cannot ignore the specification. The patentee may be its own lexicographer. The Federal Circuit Court has concluded that "It is therefore entirely appropriate for a court, when conducting claim construction, to rely heavily on the written description for guidance as to the meaning of the claims." *Philips v AWH* 415 F.3d 1303, (Fed. Cir. 2005) (*en banc*) That Smith makes a brief use of the phrase "threads of control" can not, without more, lead to a conclusion that Smith teaches an "application thread". Without more substantive description, Smith cannot be cited for teaching the claimed "application thread".

Because the combination of references fails to disclose an application thread, the claims are patentable over the cited references.

F. Claims 21-35: The References Do Not Disclose the Claimed "User Space" and "Operating System Space"

Claims 21 through 35 recite different elements in "user space" and "operating system space". However, the cited references fail disclose both "user space" and "operating system space". Further, the cited references fail to make a distinction between "user space" and "operating system space". Because the combination of references fails to disclose "user space" and "operating system space", the claims are patentable over the cited references.

G. Claims 21-25: The References Do Not Disclose the Claimed "Script Interpreter Units", "Application Thread" and "Protocol Engines" in "User Space"

Independent claim 21 recites "a plurality of script interpreter units in user space", "an application thread in user space" and "a plurality of protocol engines in user space". The Examiner asserts that Smith's user test computer 202 discloses "user space". This is not so. It is well known that a user computer includes both "user space" and "operating system space". As such, Smith does not teach "user space". Even if *in arguendo* Smith did teach "user space", Smith does not disclose the claimed "user space" as to the claimed elements. That is, Smith does not teach "a plurality of script interpreter units in user space", "an application thread in user space" and "a plurality of protocol engines in user space".

Further, Averbuj similarly fails to disclose that the sequencers of Averbuj (which are asserted to teach the claimed protocol engines) are in user space. Averbuj is silent as to whether the sequencers are in user space or operating system space. As such, Averbuj does not teach "a plurality of protocol engines in user space".

Therefore, because Smith and Averbuj fail to disclose the claimed elements in "user space", claim 21 and all claims depending thereon are patentable over the cited references.

H. Claims 26-30: The References Do Not Disclose the Claimed "Script Interpreter Units and "Application Thread" in "User Space" and the Claimed "Protocol Engines" in "Operating System Space"

Independent claim 26 recites "a plurality of script interpreter units in user space", "an application thread in user space" and "a plurality of protocol engines in operating system space". The Examiner asserts that Smith's user test computer 202 discloses "user space". This is not so. It is well known that a user computer includes both "user space" and "operating system space". As such, Smith does not teach "user space". Even if *in arguendo* Smith did teach "user space", Smith does not disclose the claimed "user space" as to the claimed elements. That is, Smith does not teach

In addition, the Examiner appears to assert that because Averbuj discloses the WINDOWS NT operating system, the sequencers of Averbuj (which are asserted to teach the claimed protocol engines) execute in operating system space. However, there is no such teaching in Averbuj. Averbuj is silent as to whether the sequencers execute in operating system space or user space.

Therefore, because Smith and Averbuj fail to disclose the recited elements in "user space" and "operating system space" as claimed, claim 26 and all claims depending thereon are patentable over the cited references.

G. Claims 31-35: The References Do Not Disclose the Claimed "Script Interpreter Units" in "User Space" and the Claimed "Application Thread" and "Protocol Engines" in "Operating System Space"

Independent claim 31 recites "a plurality of script interpreter units in user space", "an application thread in operating system space" and "a plurality of protocol engines in operating system space". The Examiner asserts that Smith's user test computer 202 discloses "user space". This is not so. It is well known that a user computer includes both "user space" and "operating system space". Importantly, Smith does not disclose the claimed "user space" as to the script interpreter units. Smith is silent as to whether the script interpreter units execute in user space.

Moreover, the Examiner fails to show where and whether Smith discloses that the threads of control disclosed in Smith (which are asserted to teach the claimed application threads) are in operating system space. The examiner has failed to make a prima facie case of obviousness because the Examiner failed to assert where Smith or another reference teaches that the application threads are in operating system space. Review of Smith shows that Smith is silent as to whether the threads of control execute in user space or operating system space. As such, Smith fails to teach that application threads are in operating system space.

In addition, the Examiner appears to assert that because Averbuj discloses the WINDOWS NT operating system, the sequencers of Averbuj (which are asserted to teach the claimed protocol engines) execute in operating system space. However, there is no such teaching in Averbuj. Averbuj is silent as to whether the sequencers execute in operating system space.

Therefore, because Smith and Averbuj fail to disclose the recited elements in "user space" and "operating system space" as claimed, claim 31 and all claims depending thereon are patentable over the cited references.

CONCLUSION AND RELIEF

In view of the foregoing, it is believed that all claims patentably define the subject invention over the prior art of record and are in condition for allowance. The undersigned requests that the Board overturn the rejection of all claims and hold that all of the claims of the above referenced application are allowable.

Respectfully submitted,

SoCal IP Law Group LLP

[illegible]

Date: August 6, 2008

Mark A. Goldstein
Reg. No. 50,759

SoCal IP Law Group LLP
310 N. Westlake Blvd., Suite 120
Westlake Village, CA 91362
Telephone: 805/230-1350
Facsimile: 805/230-1355
email: info@socalip.com

(viii) CLAIMS APPENDIX

The claims involved in this Appeal are as follows:

1. A method of creating network traffic replicating activities of a large number of users comprising:

receiving a test script including a plurality of commands

invoking a script interpreter

launching an application thread to execute the test script

invoking a protocol engine for each of the commands in the test script such that each protocol engine has an associated command,

each protocol engine executing its associated command.

2. The method of claim 1 wherein the commands in the test script simulate actions taken by a network user.

3. The method of claim 1 wherein the commands in the test script include extended operation operating system commands.

4. The method of claim 3 wherein the extended operation operating system commands include "fetch," "verify," "fetch and verify," "fetch and ignore," "monitor," and "count."

5. The method of claim 1 wherein the test script causes network traffic to be produced.

6. The method of claim 1 wherein each protocol engine executing its associated command comprises:

checking whether a maximum number of protocol engines has been exceeded

performing the executing when the maximum number of protocol engines has not been exceeded.

7. The method of claim 6 wherein the checking further comprises:

waiting for a system defined amount of time until attempting to execute again.

8. The method of claim 6 wherein the checking further comprises:

sleeping until system resources sufficient for the executing of the protocol engine are available until attempting to execute again.

9. The method of claim 1 wherein the network traffic is comprised of a plurality of data units adhering to a plurality of communications protocols.

10. The method of claim 9 wherein the plurality of communication protocols includes at least one of Ethernet, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Internet Protocol (IP), File Transfer Protocol (FTP), or Hypertext Transfer Protocol (HTTP).

11. A machine readable medium having instructions stored thereon which when executed cause a processor to perform operations comprising:

receiving a test script including a plurality of commands

invoking a script interpreter

launching an application thread to execute the test script

invoking a protocol engine for each of the commands in the test script such that each protocol engine has an associated command,

each protocol engine executing its associated command.

12. The machine readable medium of claim 11 wherein the commands in the test script simulate actions taken by a network user.

13. The machine readable medium of claim 11 wherein the commands in the test script include extended operation operating system commands.

14. The machine readable medium of claim 13 wherein the extended operation operating system commands include "fetch," "verify," "fetch and verify," "fetch and ignore," "monitor," and "count."

15. The machine readable medium of claim 11 wherein the test script causes network traffic to be produced.

16. The machine readable medium of claim 11 wherein each protocol engine executing its associated command comprises:

checking whether a maximum number of protocol engines has been exceeded

performing the executing when the maximum number of protocol engines has not been exceeded.

17. The machine readable medium of claim 16 wherein the checking further comprises:

waiting for a system defined amount of time before attempting to execute again.

18. The machine readable medium of claim 11 coupled with a network testing system.

19. The machine readable medium of claim 18 wherein the network testing system is coupled to a production network.

20. The machine readable medium of claim 19 wherein the network testing system is coupled to a test network.

21. A system to create network traffic simulating activities of a large number of users, the system comprising:

a plurality of script interpreter units in user space, each script interpreter unit to interpret a script including a plurality of commands,

an application thread in user space for each script interpreter unit

a plurality of protocol engines in user space for each application thread, each protocol engine to execute a command included in one of the scripts

an operating system in operating system space.

22. The system of claim 21 wherein the system supports a plurality of communications protocols.

23. The system of claim 22 wherein the plurality of communications protocols includes at least Ethernet, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Internet Protocol (IP), and Hypertext Transfer Protocol (HTTP).

24. The system of claim 21 wherein the operating system supports extended operations.

25. The system of claim 24 where the extended operations include "fetch and verify" and "fetch and ignore."

26. A system to create network traffic simulating activities of a large number of users, the system comprising:

a plurality of script interpreter units in user space, each script interpreter unit to interpret a script including a plurality of commands,

an application thread in user space for each script interpreter unit

a plurality of protocol engines in operating system space for each application thread, each protocol engine to execute a command included in one of the scripts

an operating system in operating system space.

27. The system of claim 26 wherein the system supports a plurality of communications protocols.

28. The system of claim 27 wherein the plurality of communications protocols include one or more of Ethernet, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Internet Protocol (IP), and Hypertext Transfer Protocol (HTTP).

29. The system of claim 26 wherein the operating system supports extended operations.

30. The system of claim 29 where the extended operations include "fetch and verify" and "fetch and ignore."

31. A system to create network traffic simulating activities of a large number of users, the system comprising:

a plurality of script interpreter units in user space, each script interpreter unit to interpret a script including a plurality of commands,

an application thread in operating system space for each script interpreter unit

a plurality of protocol engines in operating system space for each application thread, each protocol engine to execute a command included in one of the scripts an operating system in operating system space.

32. The system of claim 31 wherein the system supports a plurality of communications protocols.

33. The system of claim 32 wherein the plurality of communications protocols includes at least one of the following: Ethernet, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Internet Protocol (IP), and Hypertext Transfer Protocol (HTTP).

34. The system of claim 31 wherein the operating system supports extended operations.

35. The system of claim 34 where the extended operations include "fetch and verify" and "fetch and ignore."

(ix) EVIDENCE APPENDIX

No evidence has been submitted pursuant to §§ 1.130, 1.131, or 1.132 of this title. No other evidence has been entered by the examiner and relied upon by appellant in the appeal.

(x) RELATED PROCEEDINGS APPENDIX

Since there are no applications currently being appealed that may directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal, there are no copies of decisions rendered by a court or the Board.